International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No. (03), January 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

Design and Verification of 8-Bit RISC CPU Using Verilog HDL

Avantika Kumari[1], Kumari Amrita[2]

[1,2]B.Tech-M.Tech Student VLSI, Department of Electronics and Communication,
Jayoti Vidyapeeth Women's University (India)

ABSTRACT

RISC architecture involves and attempt to reduce execution time by simplifying the instruction set of the computer for this RISC CPU is implemented on VCS tool of Synopsys platform of Unix and the source code is written in Verilog and verification by Verilog.

Keyword : Addressing Mode, Central Processing unit , Verilog

I. INTRODUCTION

As the processor technology began to evolve beginning with 4_bit and 8_bit processors, the trend was to have more instruction capability to the processor. As a consequence, the control unit and instruction decoder became very complex. In some processors, the control unit occupies 50 to 60% of the chip area. This results in the reduction in space available to registers. Similarly, the instructions that access memory tend to slow down the code execution. This fact gave rise to RISC design philosophy that focused on a small set of frequently used instruction, thus simplifying the hardware design and improving the processor performance., [1]. In this paper Verilog is used to endorse a design and to evolve a Test bench that can reuse and it is described in step by step as defined by verification principles and methodology.

II. RISC CPU

To read an instruction the contents of Program counter are transferred to the address lines. This is done when the fetch signal is high and the address multiplexers chooses the contents of the program counter to be loaded on to the address bus. As soon as the contents of the program counter are loaded onto the address bus a memory read cycle is initiated and the instruction is read from the location pointed out by the address lines and the micro instruction code is placed onto the data bus. The program counter is incremented to point to the next micro instruction in the memory location of the control memory. The data bus transfers the micro instruction to the instruction register. The instruction register has two fields, in the different formats namely.

1. Opcode, Data operand.
2. Opcode, Address of data operand

International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No. (03), January 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

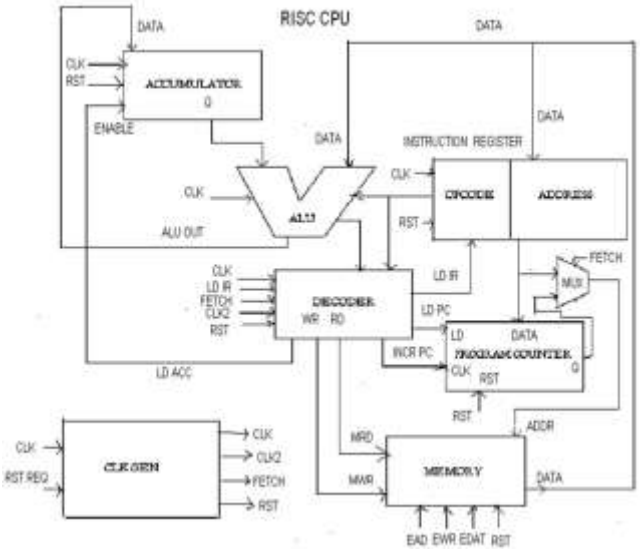

Fig 1:- Block Diagram of RISC CPU

During the first case the Opcode is given to the ALU and Decoder for decoding and a series of micro operation are generated. The data operand is loaded on to the data bus and transferred to the ALU for its respective micro operations as specified by its Opcode. In the second case the address of the data operand is loaded onto the address bus (as the fetch signal is low and the multiplexer loads the IR's address contents onto the address lines) and a memory read cycle is initiated. Here the memory location in the main memory specified by the address lines is read and the data is transferred onto the data bus and thus given the ALU to undergo the operations specified by its Opcode. The results of the ALU are stored in the Accumulator. Data operations may be combined with the memory contents and the Accumulator and result is transferred back to the Accumulator. The function of the Nor gate is that whenever all inputs are low the output is high and at all other times remains low. It is attached to tristate buffer. When the tristate buffer is enabled the data from ALU is fed to the memory thus allowing the data to be stored into the memory. When disabled the data is given to all and cut off from being written onto the data bus. Whenever there results a zero in the ALU a zero flag is set.

III. INTERNAL ARCHITECTURE DESCRIPTION

A) CLOCK GENERTOR

Clock generator generates clock, clock2, fetch signal. For every negative edge of clock, clock2 is generated and for every positive edge of clock2, fetch signal is generated. Clock2 is generated form clock and fetch is generated form clock2. They are used as the input to decoder which controls the operation of CPU. It generates reset pulse. In RISC CPU reset signal must be active-low. The reset should allow the reset of the signals to go high on falling edge of clock2 when fetch is low.

International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No. (03), January 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

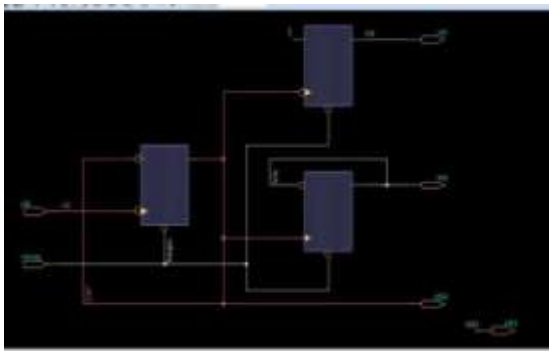

Fig 2:- RTL Design of Clock Generator

B) INSTRUCTION REGISTER

In instruction register instructions are fetched form and stored in this. It performs the action always at position edge of clock. It has 3 instruction clock, reset, load-ir and data as input and output is Opcode and address. If LDIR and reset both are high, data in instruction register splits into upper 3 bits as Opcode and lower 5 bits as q-address.

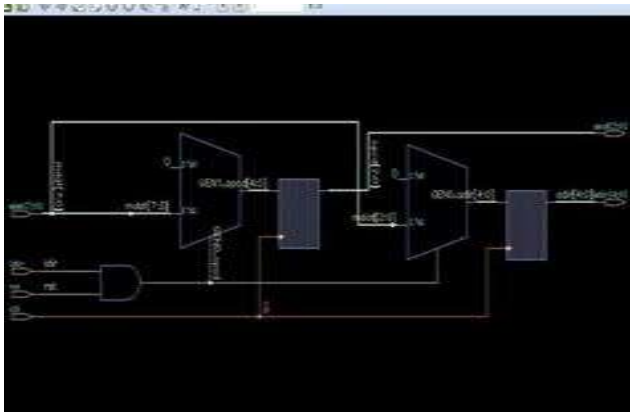

Fig 3: RTL Design of Clock Generator

C) ACCUMULATOR

Accumulator is a register; the result form Arithmetic Logic Unit is stored back in the accumulator. It has clock, reset, load-acc and ALU_OUT as input and accumulator as output. It is activated only at the positive edge of clock. If LDACC and reset both are high, data in the accumulator is loaded into ALU_OUT.

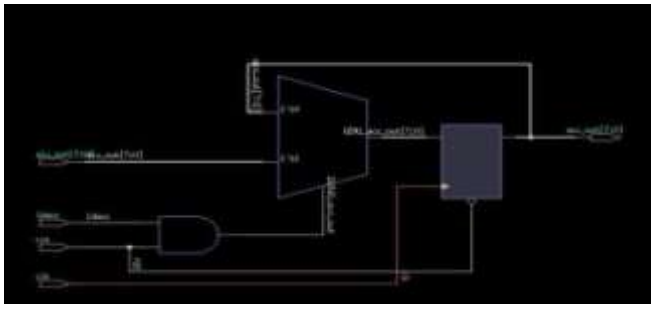

Fig 4:- RTL Design of Accumulator

International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No. (03), January 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

D) MEMORY

In RISC CPU the memory should be 8-bit wide, 32-bit location deep. Each instruction retrieved form the memory will have its upper 3 bits as the Opcode and lower 5-bit as the address. Memory block has MEM_RD, MEM_WR and address as input and data as output. If MEM_RD is high, it reads the data of memory to the data register, if MEM_WR is high, data is written to the memory.

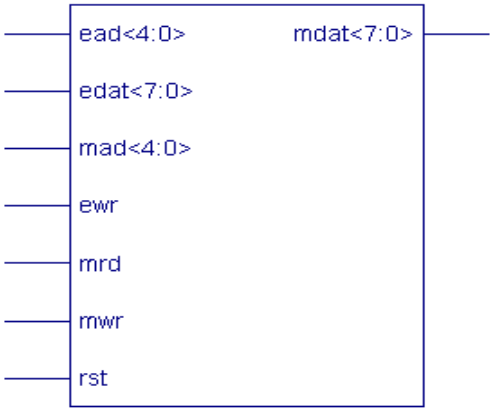

Fig 5 :- Block Diagram of Memory

E) **ARITHMETIC AND LOGIC UINT:** The ALU is a multiplexer, performs standard arithmetic and logic operations.

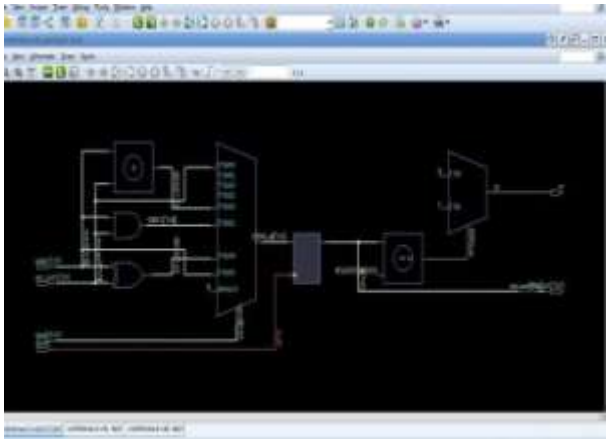

Fig 6 :- RTL Design of ALU

ALU operations should be synchronized to negative edge of the clock (period is 20). At each negative edge The operation performed is listed below a total of 8 operations performed. The ALU should perform the appropriate operation on the incoming data and accumulator, placing the result in ALU_OUT

The 3 bit Opcode decodes as follows:

International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No. (03), January 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

000	Pass Accumulator
001	Pass Accumulator
010	Add (data+ Accumulator)
011	And (data& Accumulator)
100	Xor (data ^ Accumulator)
101	Pass data
110	Pass Accumulator
111	Pass Accumulator

Table 1:- Opcode Decoding

F) MULTIPLEXER (2 TO 1):

The address multiplexer decides one output out of the two given inputs.

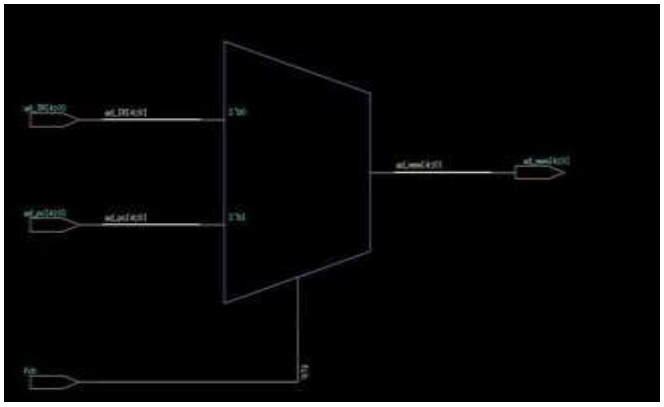

Fig 7 :- RTL Design of Multiplexer

When the fetch signal is high, the address of the program counter is transferred on to the address buses and hence the instruction is fetched. But if low the operand address specified in the address file of the instruction register transferred onto the address bus and consequently fetched.

G) PROGRAM COUNTER:

It is a 5 bit general purpose register. The program counter points to the next micro instruction to be fetched from the memory. In case of an unconditional branch the said address is loaded in to the program counter for fetching of that instruction .Normally, after the fetch cycle is completed the program counter is incremented and now those points to the next instruction.

International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No. (03), January 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

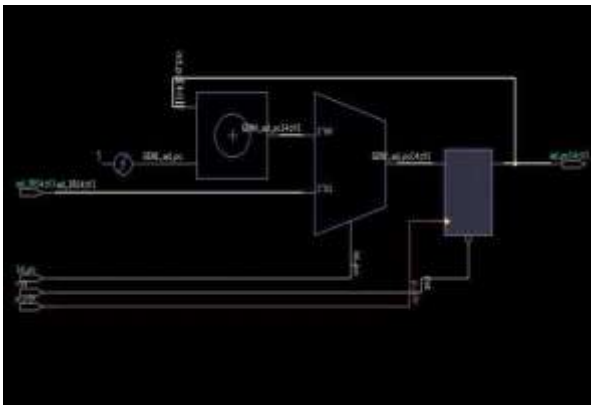

Fig 8:- RTL Design of Program Counter

H) I/O BUFFER

A buffer is a region of a physical memory storage used to temporarily store data while it is being moved from one place to another. However, a buffer may be used when moving data between process within a computer.

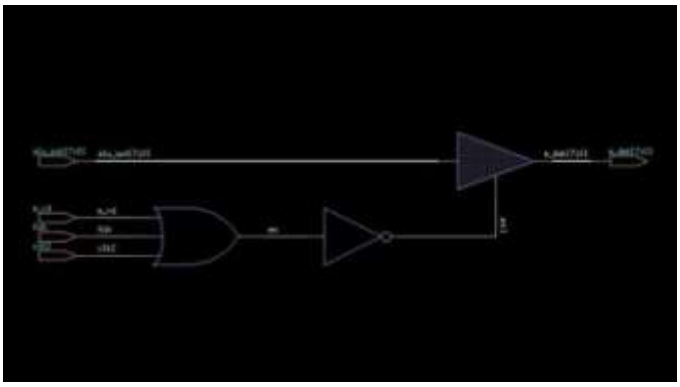

Fig 9:- RTL Design of Program Counter

I) INSTRUCTION DECODER

A decoder provides the proper sequencing of system. It has clock, fetch, clock2, reset, zero instruction and Opcode as input and LDPC, MEM_RD, MEM_WR, LDACC as output. The zero bit should be whenever the accumulator is zero. The decoder issues a series of control and timing signal. Depending on the Opcode it decodes after it receives from the instruction register.

The decoder is a simple finite state machine which consists of states. During the first states it generates a control signal for address setup. The address bus is setup and the contents of the program counter are transferred onto the address bus. The instruction fetch is generated in the second state and the instruction is read from the memory with the memory read signal and transferred onto the data bus. When the third state starts the instruction is loaded with the LDIR being high into the instruction register. The Opcode is sent to the decoder and the appropriate control and timing signals are initiated for the execution cycle. This is done in the next state in which it remains idle during the decoding of the Opcode. . The fetch cycle ends, and the program counter is incremented with the Inc signal. The execution cycle starts and the address is again setup in the fifth state, but this time instruction register's Address field is loaded onto the address bus.

International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No. (03), January 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

The operand is fetched in the sixth state with the MEM_RD signal being high and the data is transferred onto the data bus and given to the ALU for processing. In the seventh state the ALU is given its ALU_CLOCK and in synchronization with the falling edge of the clock the respective operation is performed. In the last stage the decoder issues an LDACC signal to store the result in to the accumulator.

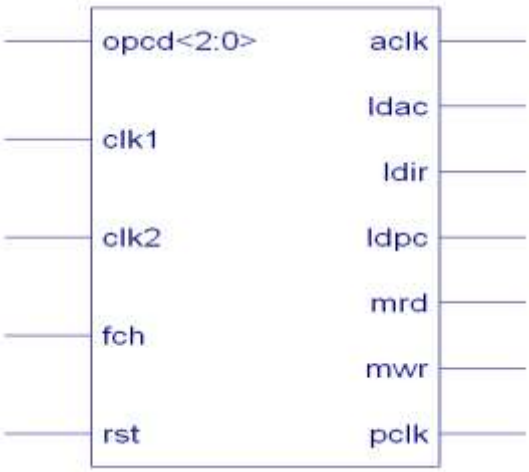

Fig 10 :- Block Diagram of Decoder

IV. VCS RESULT

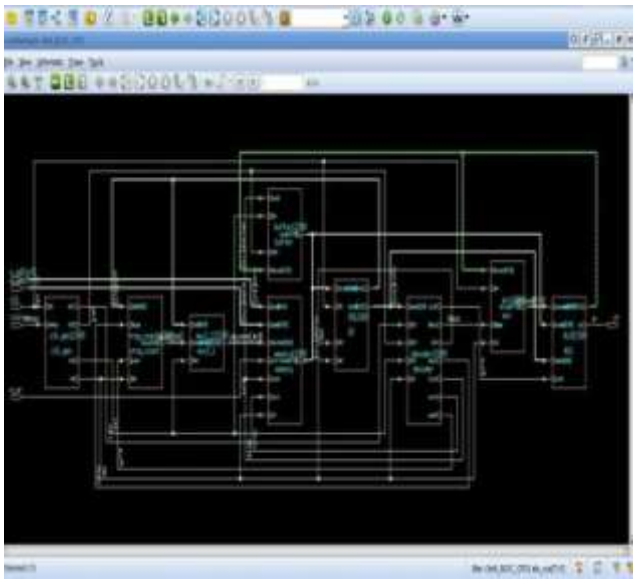

Fig 11:- RTL Design of 8-Bit RISC CPU

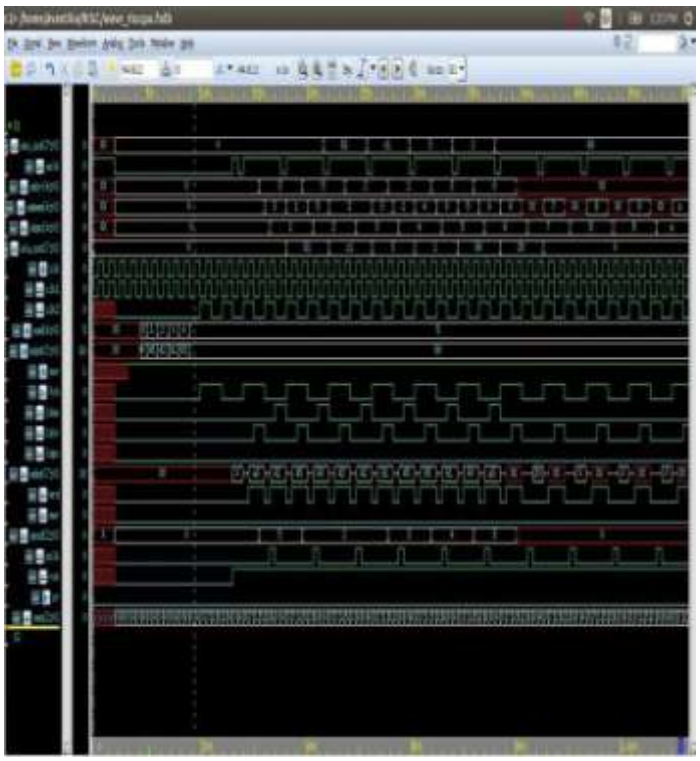

Fig 12:- Simulated waveform of 8-Bit RISC CPU

V. EDA TOOLS AND METHODOLOGIES

HDL: Verilog HVL: Verilog

Tool : VCS (Synopsys) Plateform : UNIX

VI. CONCLUSION

The design of 8-Bit RISC CPU is implemented using Verilog HDL and verified by Verilog HDL and compiled on VCS tool of Synopsys platform of Unix successfully and Addition, AND,XOR, Pass Accumulator, store operation etc. performed only in 1 cycles.

REFERANCE

[1] Shimizu, Toru, et al. "A multimedia 32 b RISC microprocessor with 16 Mb DRAM." *Solid-State Circuits Conference, 1996. Digest of Technical Papers. 42nd ISSCC., 1996 IEEE International*. IEEE, 1996.

[2] Suzuki, Kazumasa, et al. "V830R/AV: Embedded multimedia superscalar RISC processor." *IEEE Micro* 18.2 (1998): 36-47.

[3] Santhanam, Sribalan, et al. "A low-cost, 300- MHz, RISC CPU with attached media processor." IEEE Journal of Solid-State Circuits 33.11 (1998).1829-1839.

[4] Shimizu, Toru, et al. "A multimedia 32 b RISC microprocessor with 16 Mb DRAM." *Solid-State Circuits*

Conference, 1996. Digest of Technical Papers. 42nd ISSCC., 1996 IEEE International. IEEE, 1996.

[5] Sakthikumaran, Samiappa, S. Salivahanan, and VS Kanchana Bhaaskaran. "16-Bit RISC processor design for convolution application." Recent Trends in Information Technology (ICRTIT), 2011 International Conference on. IEEE,2011.